JACG

NEWSLETTER
Vol.3 No.2

Oct. 1983

Single Copy
Price $1.50

THE JERSEY ATARI COMPUTER GROUP

Special Topic: Programming

From the Editor's Desk...

Here it is. The October issue of the JACG Newsletter. This is a landmark issue. We are into our third year as a bonafide User Group with the best Atari Newsletter in the country. That is something we can all be proud of.

This is the special programming issue I have promised for several months. Finances and time have delayed but not cancelled what is our finest Newsletter yet.

Finally, this is a landmark issue for me. I am resigning as Editor of the JACG Newsletter to pursue other Atari-related interests. I will continue to contribute to the Newsletter on a regular basis and you can find me in Creative Computing Magazine, manning the Outpost column.

I have nurtured the Newsletter over the last 14 months through good and bad times. It has been an extremely rewarding experience for me and I am proud of the result. I think I have grown during the experience and learned a great deal about publishing.

As of this writing, my successor has not been chosen. However, whoever it is, will be carrying on in the tradition that was started by Dick Kushner almost two years ago. I wish the new Editor the best of luck.

L. Arthur Leyenberger

Arthur Leyenberger
Editor in Chief, JACG Newsletter

IN THIS ISSUE

AND NOW A WORD FROM THE PRESIDENT

I want to thank Ned Hancock of 3M Corp. for coming to our September meeting to discuss the production and handling of floppy disks and for giving the group 25 boxes of disks to use in a drawing for members. We gave away 15 boxes at the meeting and will give away the remainder at the October meeting.

At the Oct. meeting, when this newsletter is distributed, we will be electing officers for the coming year. Several of the officers will be continuing in their current capacities and three will be leaving. Dennis Kushler, our V.P. since our formation, will be moving to Michigan (JACG's loss is MACE's gain). Dave Logothetis, who ably filled in as Secretary when Ed Picciuti resigned, has decided to move on to another computer other than the Atari. Art Leyenberger, our very talented newsletter editor, will be leaving that post and taking over the Outpost:Atari column in Creative Computing as well as doing other writing (for JACG and other publications).

I want to thank Dennis, Dave and Art for doing a great piece of work in helping to make JACG as good as it is.

Art and I will serve as a committee of two to discuss the newsletter editor job with two members who are interested in the position. I have invoked my executive power to decide that the editor should not be an elective office, but rather one that is filled by consultation and discussion involving past editors. I hope that in this way we will get the best possible person (or persons) to fill this most important slot.

The election of officers to be held at our October meeting will have the following candidates:

President	: Dick Kushner
Vice President	: Scott Brause
	Dennis Hoskins
	Curt Springstead
Secretary	: Anne Tom
Treasurer	: Rick Olson

In addition, we have the following members on the executive committee:

Nwsltr Editor	: Frank Pazel
Editor at Large	: Art Leyenberger
Librarian	: Don Ursem
Prog. Chrm.	: Rich Rospond
BBS Sysop	: Scott Brause
Advertising	: Herb Lehner

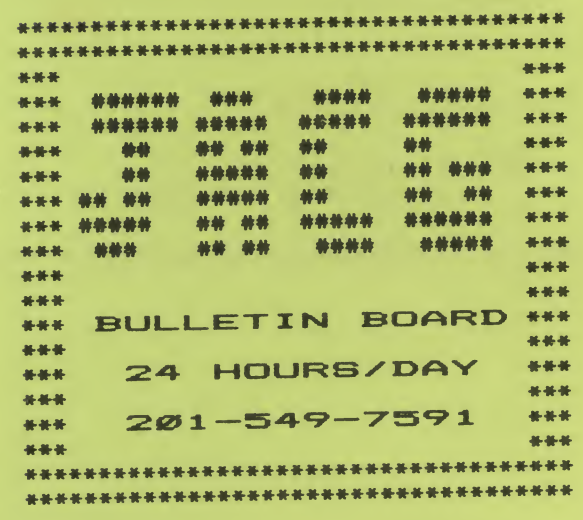
As you can see, the only contested post is for Vice President. In the absence of by-laws, the only election to occur at the October meeting will be for Vice President.

I have,in addition, created a new position for JACG, that of Editor-at-Large, and have appointed Art Leyenberger to that position. The job of this officer is to write a column of software reviews for the newsletter and demonstrate software at the meetings, all in a humorous manner. I think that all will agree that Art certainly fills this bill.

I would be remiss if I did not add the following very personal comments. Over the last year of working closely with Art Leyenberger, I have seen the tremendous dedication he brought to the job of editor. This is clearly reflected in the quality of the product. We have the best Atari newsletter in the country, bar none, and Art can take a lot of the credit and, I hope, some measure of pride in that accomplishment. Dennis Kushler is the main reason that we were able to get the use of the Bell Labs auditorium. If he did nothing else he would have been a valuable asset, but he has done much more than that. Many is the time he has gotten us out of a crisis when equipment failed to operate properly at meetings or when yours truly forgot to bring something. We'll miss his behind the scenes assistance.

One last comment on our newsletter. We are developing a stable of very talented columnists who will contribute on a regular basis. Please don't take this as an excuse to not contribute to the newsletter yourself. We need and can use all the input we can get. The more different perspectives we can bring to our publicaton the more all of our members will profit from it. In other words, keep those programs and articles coming!!

Richard Kushner
JACG President



TIDBITS

News and Views By
Arthur Leyenberger - JACG

RANA, RANA, again and again. The RANA drives finally came out. Way ahead of the schedule mentioned as a rumor last month. However, they were not without some minor problems, which have already been fixed.

One problem was the inability of a RANA drive to play second fiddle, so to speak. It could not act as drive #2-#4. Had to be #1. A new PROM fixed that egocentric problem mighty fast.

The other problem supposedly resulted from Atari's sloppy 810 drive construction. Bear with me, I'll try to explain it in 25 words or less. Seems the 'ol 810 drive was a poor design (so goes the story) in that when the head encountered a bad sector it kept on reading. Other drives - Apple, IBM. etc. - would commit suicide in this situation. Consequently, the Atari software vendors took advantage of this and designed some elaborate protection schemes such as bad sectoring, etc.

The punch line is that Percom, RANA and even the Atari 1050 had to be less than state of the art in order to be compatible with the existing software. That is, the drive manufacturers have had, and are still having, trouble designing an inferior disk drive. How's that for a strange story. See the October COMPUTE!, Insight Atari column for some more incredible info about drives.

Cassette Tape Conglomerate to Buy Out ATARI?

Hey! Really. It was printed as a rumor in John Dvorak's column in a recent InfoWorld. North American Phillips is supposedly in the process of buying Atari from old man Warner. Seems the deal was that Atari would 1) lay off a whole bunch of white and blue collar folk; 2) move manufacturing offshore; 3) Come up with a new product line. All of this has come to pass. Let's see if the magic Sept. 30, 1983 comes and goes without a new moniker on our machines.

Hey! Dvorak said it was only a rumor. Big JD likes to spread rumors and innuendos. Hey! Really!

What the Hell is Going On?

My sources tell me that in the New York Metro area there is, and has been for quite a while, a shortage of Atari product. For a while, 850 interfaces where hard to get. Then drives. Now programmer kits, drives and 800s. However, in other parts of the U.S.A. there ain't no shortage at all. Why the discrepancy? Is there something funny going on with the local distributor? Is there something funny going on with Atari?

Rumor has it that Atari is buddy- buddy with west coast retailers but require local retialers to pay for parts *in advance* rather than sending an urgently needed part.

These kind of practices, if true, have the effect of turning away prospective Atari computer purchasers. Also, if the retailers cannot get the product, they will soon abandon Atari for some computer maker that's got their act together.

This is an open invitation for Atari and/or Chancellor (the local distributor) to respond to these comments right here in this newsletter. I will print any response I receive.

What the hell is going on?

Another DOS Utility

Last month I mentioned the useful sector copying utility, SCOPY 810. Another useful DOS utility I use and one that I would recommend to anyone who does a lot of file manipulation is *DOS-MOD* from Eclipse software. *DOS-MOD* is completely compatible with your existing Atari DOS 2.0S and contains a number of enhancements.

How would you like to be able to use full screen editing when using DOS, just like you do when using Atari Basic? You got it. How about being able to see more of what you are doing on the screen? No problem, *DOS-MOD* gives you more than half the screen since its compressed menu takes up less room. Also, there is a minimum of screen clearing.

If this was all that *DOS-MOD* allowed me to do I would say, "ok, but so what?". There's more. Commands that used to take several lines and require answering prompts can now all be put on one line. The wildcard operators now work(as they should) with COPY, DELETE and RENAME. And get this, *DOS-MOD* lets you create command files which may contain a sequence of commands that will be executed in one operation.

Still not convinced? OK, throw in an excellent set of on-line tutorials that, when copied to a printer using "C D1:*.*,P:", yield 48 pages of documentation. Finally, *DOS-MOD* fixes many of the bugs in Atari DOS such as the RS-232 handler being destroyed on RESET and typical MEM.SAV problems.

DOS-MOD is an excellent product that greatly simplifies and expands the Atari DOS into a powerful operating system. It is available from: Eclipse Software, 1058-J Marigold Court, Sunnyvale, CA 94086. It costs $35.00 and comes with a 30-day money back guarantee. ■

THE AUSTIN FRANKLIN 80 COLUMN BOARD
A HANDS-ON REVIEW
by Dick Kushner-JACG

You have read in past issues of this newsletter about the trials and tribulations of trying to get one of these boards to evaluate. Well, one has finally arrived! We needn't dwell on all the past frustrations, but rather try to give the product a fair evaluation based on its performance.

The Austin Franklin 80 column board fits in the last memory slot of the 800 and is jam packed with electronics, with most of the chips having the identifications scratched off (take that, you unscrupulous pirates!). There is a cable that attaches to one of the two connectors available on the board and which runs out the side where it divides to go to the monitor jack on the 800 and the input of your monitor. You, of course, need a monitor to support the resolution required for an 80 column display. There is a third connection available in order to input a lightpen, with this connection being exactly like one of the joystick inputs. Because the board goes in slot #3, you must either reduce your memory to 2-16K boards or get a 32K or 48K board to get you back to 48K in the remaining two memory slots. I am using an Austin Franklin 48K board in slot #1 with a loop-back board in slot #2 (in order to trick the computer into accepting all 48K in one slot). The board also comes with a cartridge that plugs into the right cartridge slot when you want 80 columns. If you want the usual 40 columns, you leave the board in the computer and just remove the right cartridge.

If you boot BASIC while in the 80 column mode, you are greeted with the usual READY prompt and a blinking cursor. You always have a blinking cursor while in the 80 column mode. There are a number of commands that you can send to the board from your keyboard or from within a program. For those with RGBI monitors, you can use the following:

CTRL+F	sets all RGBI guns on
CTRL+E	turns red gun off
CTRL+N	turns green gun off
CTRL+L	turns blue gun off

Since I don't have access to a RGBI monitor, I cannot report firsthand experience with this mode, but Austin Franklin gave a very impressive display of this multicolor mode at our September JACG meeting

The vast majority of users will not have a RGBI monitor available, since they cost $400-$700. For users of monochrome monitors, which cost from $80 to $250, you can use the following commands:

CTRL+O	to enable the following:
CTRL+B	toggle blink
CTRL+I	toggle intensity
CTRL+U	toggle underline
CTRL+P	toggle 78 column beep
CTRL+R	reset all toggles
Atari key	toggle inverse video

To use these commands from within a program, simply precede the appropriate code by pressing the ESC key once. The only exception to this is that use of the Atari key need not be preceded by ESC. All the toggle commands turn the feature ("attribute") on the first time they are invoked and off the next time. All attributes are in the off state when you first bring up the system. The blink toggle will make all text typed while in this mode blink on and off at the same rate as the cursor. Cancelling this toggle means that anything added to the screen will then not blink. However, anything that was blinking will continue to do so. The same on/off capability holds true for the other features that toggle.

The possibilities for using these features are intriguing. In a word processor, you might see your underlined text on the screen as well as on the printed document. Italics might appear in blinking text and bold printing might be displayed on the screen as brighter characters. Alas, these are all daydreams at the moment because there is no software currently available that makes use of this 80 column board other than a terminal program from Austin Franklin. I have learned that LJK will be developing versions of Letter Perfect and Data Perfect to work with this board and that Synapse will have compatible versions of SYN-TEXT, SYN-FILE, etc.

The initial Austin Franklin software cartridge did have a couple of serious limitations. The "line insert" command were quite slow, especially when compared to all the other editing commands (i.e., character insertion/deletion and line deletion). Also, the SYSTEM RESET was not properly handled. These problems were fixed and the character set was also improved before many units went to dealers. The manufacturer responded promptly and expertly to the problem. This bodes well for future support.

So where does the Austin Franklin 80 column board stand? It appears to be a well designed product that will meet a need in the Atari community if we soon see software that takes advantage of its capabilities. Until then, it is sort of like adding a unit to your car so that it will run on fusion energy - the promise is there, but not the reduction to usefulness. I eagerly await the chance to test the Austin Franklin 80 column board with compatible software.

●

KEYPUNCH CARDS

The Datasoft Atari BASIC Compiler
Reviewed by Charles Scala

First, there was Man. And he was good. Unfortunately, he was too slow. So Man created a calculating machine. And it was good. Unfortunately, it was not efficient or fast enough. So Man created the computer. Unfortunately, computer languages were not easy enough to be readily utilized by the general public. So Man created BASIC. And it was good. And efficient. Unfortunately, it was too slow (?). So Man decided to quit while he was behind and start working on a better television set... Meanwhile, Datasoft (actually Phillip Dennis of Special Software Systems for Datasoft) released "The Atari BASIC Compiler". And it is good, efficient, easy, and fast!! [Ed. Note: There you have it, the entire history of human evolution!]

For those of you who would really like to see your best programming efforts whiz by, or if you put off writing something because you felt that Atari BASIC might not be fast enough, this may be for you. With a few guidelines (restrictions?) to follow, almost any Atari BASIC program can be easily converted to high speed code. I was skeptical about the 'ease of use' until I read the 37 page manual (written so humans can understand it). How easy is it? Try this: 1] Create a work disk for the compiler. 2] Load up the compiler and tell it what the name of your BASIC file is and what name you would like to give to the compiled version. 3] Then tell it if you would like the standard floating point or the integer math package (more on that later). 4] Insert the disk(s) and it goes off on its own. How's that? The "work disk" consists of a few files that you are allowed to copy from the master disk and your program file. A two drive system will compile using two disks. If you have one drive (guess what?) you have to switch disks about five times during the compilation process (I know how you feel about swapping disks. But don't worry - Man is working on it...).

Yes, I said integer routines! If you pick that option, all fractional math will be treated as if there was INT() around everything. The integer math package does execute faster - A blessing to the game writers and number crunchers out there. The chart following has all (?) of the practical combinations that I could think of in a benchmark test (Benchmark programs used are from PPC Calculator Journal, V9,N5). Don't tell me you actually thought there wouldn't be any problems with this? One that I found is that the FASTCHIP (OS math chip) will not function properly using the FP routines (the INT seems to work fine). That is why that column does not exist:

	ATARI BASIC	BAS FC	FP	INT
B1:	2.5	2.3	.9	.3
B2:	8.0	7.0	1.0	.3
B3:	21.3	19.7	8.9	1.7
B4:	25.0	19.0	11.9	.9
B5:	28.7	21.7	12.0	1.4
B6:	43.6	36.7	16.2	2.2
B7:	1:06.1	56.3	23.0	3.3

Across the top are : Atari Basic cartridge, Atari BASIC with a FASTCHIP installed, BASIC compiler with FP routines, and BASIC compiler with INT routines. They were timed with the stopwatch - eyeball method so please bare with the expected degree of error. Here are the programs I used:

B1:

```
300 PRINT "START"
400 FOR K=1 TO 1000
500 NEXT K
700 PRINT "END"
800 END
```

B2:

```
300 PRINT "START"
400 K=0
500 K=K+1
600 IF K<1000 THEN 500
700 PRINT "END"

800 END
```

B3:

```
300 PRINT "START"
400 K=0
500 K=K+1
510 A=K/K*K+K-K

600 IF K<1000 THEN 500
700 PRINT "END"
800 END
```

B4:

```
300 PRINT "START"
400 K=0
500 K=K+1
510 A=K/2*3+4-5
600 IF K<1000 THEN 500
700 PRINT "END"
800 END
```

B5:

```
300 PRINT "START"
400 K=0
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB 820
600 IF K<1000 THEN
    500
700 PRINT "END"
800 END
820 RETURN
```

B6:

```
300 PRINT "START"

400 K=0
430 DIM M(5)
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB 820
530 FOR L=1 TO 5
540 NEXT L
600 IF K<1000
    THEN 500
700 PRINT "END"
800 END
820 RETURN
```

B7:

```
300 PRINT "START"
400 K=0
430 DIM M(5)
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB 820
530 FOR L=1 TO 5
535 M(L)=A
540 NEXT L
600 IF K<1000

    500
700 PRINT "END"
800 END
820 RETURN
```

Continued on 24

★ Art's Arcade ★

By Arthur Leyenberger - JACG

Welcome back to Art's Arcade. It seems like we last talked just a short time ago. Anyway, this month I am doing something a little different. I have four games to talk about, 2 each from two different software houses. Although I discussed one game from each of them last month, I felt that you would want to know about their latest efforts. So, this month, Worms? and Murder on the Zinderneuf from Electronic Arts and Super Cobra and Q-Bert from Parker Brothers will be reviewed.

Before I start, I must tell you that this is being written on a beautiful late summer evening at Cape Cod. This is my first vacation here and I can see why once you experience it, you will never want to leave. Beautiful weather, balmy breezes and rugged landscape will cause the Jersey shore to never again be the same for me. *On to* business.

Worms? by Electronic Arts

I am not sure how best to describe Worms?. It's not a shoot-'em-up. It's not an arcade game. Maybe an adventure? Well, not really. "Come on, Art, what is it." The closest I have come to understanding what Worms? is (aside from playing it for hours) is to quote the liner notes (yes I said liner notes. Like all Electronic Arts games the packaging is similar to an LP - liner notes, artist biographies and beautiful artwork.)

"[Worms is] a way to think about thinking. Imagine a new kind of creature, animated but not alive. Synthetic, yet intelligent. A product of modern electrogenetic science. Imagine that these creatures live in a world where reality is a two-dimensional grid. Without material bodies, they exist as light: pure energy. Without politics or religion, their lives are an unending adventure in pure mathematics, in music, in geometry. And now, suppose they are worms."

Wow! Are we talking video games here or what? Did that quote give you a clue to what Worms? is? No? that's ok. It sounds more like existential claptrap. But remember, Electronic Arts was the company who advertised, new wave style, "Can your computer make you laugh, can it make you cry?"

The graphics are very, very pretty. The music adds to the experience. If you can imagine a very bizarre version of Qix then you are close to understanding. But instead of simply controlling the movement of the lines with your joystick, you actually program the worms to move. You program (teach) the worms to move in a particular pattern in order to capture territories by laying trails from dot to dot. Each dot lies in the center of a territory and a point is earned by having your worm lay the last trail in a terrirtory.

Sound confusing? It really isn't. The game can be played on an intellectual level or any level you want. It's challenging and fun. It can even be used for relaxation therapy.

The artist responsible for this mind-expanding game is David Maynard. Worms? costs $40.00, requires 32K disk and is unique.

Murder on the Zinderneuf
By Electronic Arts

It is 1936. You are 5000 feet over the Atlantic Ocean flying in the luxury dirigible Zinderneuf. Sixteen pasengers were aboard the ship when it left London. Twelve hours out of New York, only fifteen remain. The crime is murder. The victim can be anyone (in each game). The fifteen passengers still alive are the suspects. You are the detective.

So starts the mystery adventure: Murder on the Zinderneuf. Like a classic mystery, Zinderneuf has all the elements - plot, counterplot, clues, suspects and a famous detective to sort things out and solve the crime.

As each game begins, you choose from eight different detectives. With tongue in cheek, some are reminiscent of famous detectives: Inspector Klutzeau, Lt. Cincinnato, Agatha Marbles. You get the idea. Your choice determines how quickly you will find clues, how explicit the clues are and how (and how well) you interrogate the suspects.

You roam freely throughout the dirigible by scrolling with your joystick. You may follow suspects and if desired you may interrogate them. The questions you ask are displayed at the top of the screen. Also, depending upon which detective's identity you have assumed, you may ask the questions in a variety of ways such as forceful, naive, pushy or polite. The manner in which you ask a question will partly determine the answer you get.

You can also search the rooms of the suspects for clues. Once you think you have enough information and clues, you may accuse a suspect. If you are correct and the suspect believes you have enough evidence, he or she will confess, the game will end and you will receive a rating from feeble flatfoot to super sleuth. If you are wrong or don't have enough evidence, the suspect will refuse to speak with you for the remainder of the game. And it won't do your reputation, and rating, much good either.

Zinderneuf was created by Robert Leyland, Paul Reiche III and Jon Freeman. It costs $40.00, requires 32K disk and is an enjoyable participative mystery that, unlike regular text adventures, does not become worthless once you solve the crime. A new victim is chosen each game and you can assume the role of any of the eight detectives. Zinderneuf is especially fun when several people get together and try to solve the mystery.

Q-Bert by Parker Brothers

Parker Brothers is coming on strong. Their first two Atari computer game products, Astro Chase and Frogger, were previous hits. Now they have come out with two arcade classics: Q-Bert and Super Cobra. At this rate, Parker Brothers will become a formidable competitor for your game dollar.

Continued on 23

Getting down to BASICs
by Dick Kushner-JACG

It is more than a little embarassing to make a major mistake in the very first column (last month). I hope that some of you picked it up. In the beginner's section I said that in the deferred mode of execution, commands are only carried out when you press <RETURN>. Horror of horrors, what a faux pas. Of course we all know (or will after this month) that when a line is preceded by a number, pressing <RETURN> stores that line in the computer's memory and it's instructions will only be carried out when you type RUN. By the way, line numbers in programs may have any integer values from 0 to 32767. Also, in the intermediate section, I failed to mention that the "chip" referred to is the GTIA chip. I'll try to avoid such goofs in the future.

YOU BOUGHT AN ATARI, HUH? NOW WHAT?

Here again is our first program:

```
100 PRINT "THIS IS BEING TYPED"
110 PRINT "ON THE ATARI HOME"
120 PRINT "COMPUTER."
```

What else can we learn from this program? We used RUN without preceding it with a line number to cause this program to be executed. The result was:

```
THIS IS BEING TYPED
ON THE ATARI HOME
COMPUTER.
```

The PRINT commands results in printing exactly what is contained within the quotation marks. The lines are printed one underneath the other; that is, after each line is printed the cursor moves down one line and back to the beginning of the line before printing the next line. When the computer ran out of program lines it skipped one line, printed READY and placed a white block at the start of the next line. This white block is the cursor and indicates where the next character typed will appear.

What if we only wanted to print the last line of the message? Use GOTO 120 and press <RETURN>. You will get COMPUTER. printed on the screen and then the READY prompt. Typing RUN will result in the entire message again. Once the program is stored in the computer's memory it will remain there unless you change it, erase it by typing NEW or turn off the power. To get a display of the program, type LIST and press <RETURN>. The computer will dutifully list the current program stored in memory onto the screen. If, while the program is being listed on the screen, you press CTRL and 1 at the same time the listing will halt. Pressing them again will continue the listing.

That's quite a few new concepts for one sessions. We'll take up this discussion here next month.

THERE'S MORE TO BASIC THAN THE *BASICs*

We began to look at GRAPHICS 9, 10 and 11 (the so-called GTIA modes) last month. Now let's concentrate on GRAPHICS 9. This mode allows 16 shades of one color. With this many shades, very realistic shading effects can be displayed. You select the color with

```
SETCOLOR 4,B,0
```

where B is the color value (0-15). The brightness is then selected with

```
COLOR N
```

where N is 0-15. Notice that, unlike other graphics modes, in GRAPHICS 9 the COLOR command selected a brightness. The following program will display the shadings available.

```
90 REM * GRAPHICS 9 SHADING
100 GRAPHICS 9:SETCOLOR 4,1,0
120 X=50:FOR I=0 TO 15:COLOR I
130 PLOT X+I,0
140 DRAWTO X+I,100
150 NEXT I
160 GOTO 160
```

This only gives an inkling of what is possible. Let's develop a program that will do the following:

1. Draw a star field as a background
2. Draw a monolith as in "2001"
3. Cycle through all the possible colors

We'll use the RND function to plot 500 "stars" randomly on the screen. We can then use a routine much like the one above to draw the monolith. To get a depth effect we will continuously vary the height of the drawing and the shading. For a more blocky appearance, we'll draw a front face and shade it in the opposite direction. The result will be a 3-D monolith. Next we will have a loop that uses SETCOLOR 4,C,0 to change the color by changing the value of C. Here's the program that does the job. All that's missing is the familiar "2001" theme. However, music will have to wait for another column. We'll have some more to say about our current program next month.

```
90 REM * 2001 MONOLITH + STARS
100 GRAPHICS 9:SETCOLOR 4,0,0
110 YTOP=35:YBOT=155
112 GOSUB 210:REM * DRAW STARFIELD
115 C=4:FOR I=46 TO 49:COLOR C
116 PLOT I,19:DRAWTO I,171
117 C=C+1:NEXT I
120 X=30:FOR I=15 TO 2 STEP -1:COLOR I
130 PLOT X+I,YTOP-I
140 DRAWTO X+I,YBOT+I
150 NEXT I
155 FOR DELAY=1 TO 1000:NEXT DELAY
160 FOR J=1 TO 5
170 FOR I=0 TO 15
180 SETCOLOR 4,I,0
185 FOR DELAY=1 TO 50:NEXT DELAY
190 NEXT I
200 NEXT J
205 SETCOLOR 4,0,0:GOTO 205
208 REM * 500 RANDOM STARS
210 SETCOLOR 4,0,0:COLOR 15
220 FOR I=1 TO 500
230 X=INT(RND(0)*79+1)
240 Y=INT(RND(0)*191+1)
250 PLOT X,Y
260 NEXT I
265 FOR DELAY=1 TO 1000:NEXT DELAY
270 RETURN
```

▲

The Report Card

By Frank Pazel - JACG

The most exciting piece of educational software I have seen for the Atari has just made its appearance. It contains things like Demons, Garbage Collectors, Toots, and Turtles. It is not an arcarde game but I've been using it for some weeks now with a group of eighth grade youngsters who can hardly contain themselves until it appears on their terminal screen.

It is Atari's new language cartridge, Logo. To say that it met all of my expectations and hopes is truly an understatement. Developed for Atari by Logo Computer Systems, Inc. of Lachine, Quebec, Canada the Logo package is superb. It comes with a 16K left-insert cartridge, reference manual, introductory turtle graphics manual, and quick reference guide. Although not intended to replace PILOT it seems destined to put a real dent in the sales and enthusiasm curves of that language.

Logo, as you may know, was developed over ten years ago at M.I.T. by Dr. Seymour Papert in an attempt to make the programming task more of an English language, common sense process. The success of his experiments culminated in what can be best described as a near-cultist following of Logo enthusiasts. After a few hours with the program and many hours watching 14 year olds take to it painlessly I am just about to join the fanatics' ranks.

The graphics mode of Logo uses the image of a small turtle (which looks like a cute little creeper - not a triangle or X as in some computers' versions) to move about the screen with extremely simple commands. FORWARD 50 moves the turtle with un-turtle like rapidity 50 spaces forward on the tv screen. RIGHT TURN 90 makes him (her? it?) turn 90 degrees to the right. Combine these two to quickly draw a square, hexagon, or whatever. You should see beginners get into the graphics patterns after learning just these two simple commands. There are several pages of commands in the Reference Guide. Make no mistake; Logo is not a "kiddies" language. It is a powerful, full blown lexicon with few constraints.

There is a text window, which you can remove with two keystrokes, a text-only mode, and all kinds of colors and sounds (TOOTS) which are easily incorporated into your program. Programs are written by typing EDIT and being transported automatically into the Atari Logo Editor. There are no line numbers to worry about and if the program has a bug in it Logo tells you so in polite, plain English (YOU DON'T SAY WHAT TO DO WITH XXXX, I DON'T KNOW HOW TO XXXX, etc.). You can save to and load from cassette or disk, dump to printer, and perform just about every housekeeping function you might want. The kids love the idea that the programs can be named for themselves and called up in

Continued on 23

ATARI Programming Languages

By Rich Rospond - JACG

Last month we looked at programming languages in general, and BASIC in particular. This month we will look at Pascal so you can see how it differs from BASIC.

Niklaus Wirth of Zurich, Switzerland invented Pascal in 1971 as a tool for teaching ALGOL, and to demonstrate the principles of astructured language. The language was named for Blaise Pascal, the French mathematician who invented one of the first mechanical computing devices.

Pascal compilers were written for mainframe computers and the language has risen in popularity. At the University of California, San Diego, Pascal was implemented on mini and micro computers. The result has been the UCSD Pascal system. This is not only an implementation of Pascal, but an entire operating system that includes several editors, a file handler, assembler, compiler and debugger. UCSD Pascal now runs on the most popular home computers.

The ATARI Pascal language system is not UCSD, requires 2 disk drives and 48K. One disk drive holds the Pascal compiler, and the other disk holds the editor in which your code is assembled.

Like BASIC, Pascal uses English words and the conventional math symbols. It operates on standard data types such as integer, real, and boolean while giving the programmer the freedom to define new data types. You can also define new functions and procedures.

Pascal is a compiled language, but it does not usually compile into machine code. Instead it compiles into an intermediate pseudo-code called P-code. The p-code is saved on a disk file, then interpreted into themachine code of the computer.

The purpose of p-code is to make the language portable. To move Pascal to a new computer, you have to write a new intepreter from p-code to the machine language. This is far simpler than adapting a complete language. The p-code can be the machine language of the processor, allowing it to run without the interpretive step.

Statements in Pascal are separated by semicolons instead of line numbers, and can be written free format. That means that the computer will read a statement placed anywhere on a line, or several lines. It is also block structured , meaning that the programs are composed of blocks each having a BEGIN and an END. Each block or paragraph is in effect an independent or sub program. Pascal is highly structured, and flows logically from beginning to end without the abrupt shifts that characterize numbered programs such as BASIC.

Pascal allows the user to define and manipulate new types ot data other that

Continued on 21

THINGS THE MICROSOFT MANUAL DIDN'T TELL YOU

by Frank C. Jones, DC Group
Reprinted from CURRENT NOTES, 3/83
Re-typed by Celia Fletcher

What's Wrong with the Manual?

I recently read a review of Atari Microsoft BASIC that gave it top marks in all departments. The reviewer especially singled out the manual for praise, stating that it was the time required to do a good, careful job on the documentation that held up the release of the language. I have had Microsoft BASIC for several months now and, while I am very happy with the language itself, I must disagree with that reviewer about the manual that comes with it. While a lot of the documentation is good and complete, some of it is incomplete (it raises more questions than it answers), some is hard to understand, and, worst of all, some is just plain wrong.

While I worked with the language I kept a notebook of the things I found out by poking around (no pun intended). While I am sure that these notes are also not complete, I believe that they are correct and, since the flow of discovery had slowed to a trickle, I thought it was time to write them up for whatever benefit they may be to others. So here is what I have found up to now.

Numbers

As many of you may know, numbers come in three varieties in Atari Microsoft BASIC: single precision, double precision and integers. Here are some quirks to watch out for. Numbers between -32768 and 32767 can be stored as integers, provided they have no fractional part. If a literal has a decimal point, it is considered to have a fractional part, even though no digits follow the point. Thus, 25 is an integer but 25. would be treated as a single-precision real. Also, arithmetic involving only integers is integer arithmetic yielding truncated results, so the result of 1/2 is 0 but 1./2 or 1/2. would yield 0.5 for an answer.

Double precision numbers are accurate to sixteen decimal places; however, they don't always look that accurate. If you construct a number that is an integer plus a small increment, the incremental part will not print correctly if it is too small. For instance, if you add 1 and 1D-06 and then print the result, you will see 1.000000005 instead of the correct number, 1.00000001; the incremental part appears to have been divided by two. If the increment is smaller yet, say 1D-10, it will appear to be divided by four, and so on.

Things are not as bad as they seem. Subtract 1 from the "wrong" numbers and print the result. You will recover the correct incremental value that you added in the first place; it was there all the time. What appears to be wrong is not the arithmetical operations but rather the conversion to ASCII form during the printing operation. Disconcerting but not fatal.

One more thing about numbers: integers may be entered in hexadecimal form but they will print in decimal as signed, two-byte numbers. That means the number you enter as &FFFF will be considered as and print as -1.

Control Structures

Microsoft has something ELSE. The IF ... THEN structure has been enlarged to include IF ... THEN ...ELSE. This works as described in the manual; you may have more than one statement where the dots are, but it all has to fit on one line (logical line, of course).

The WAIT command is described incorrectly in the manual. It has the form

```
WAIT address,AND_mask_byte,compare_to_byte
```

and the manual states that the mask is aplied to the compare_to_byte. This not only makes no sense, it is not correct; the mask is applied to the value in the specified address before comparing it to the compare_to_byte as it should be.

Unlike Atari 8K BASIC, there may be only one NEXT statement for each FOR statement. This holds even if they are on mutually exclusive branches of the program. The first one is the one that is considered valid; if any others are accessed by the program, it will cause an error.

Contrary to what the manual says, a subroutine RETURN statement returns you to the statement following the GOSUB, not to the line following it.

The AFTER statement expects a RESUME statement, just as ON ERROR does. Furthermore, the END statement does NOT reset the time stack as is claimed in the manual. A line such as

```
AFTER(360)100:END
```

will cause the program to pause for six seconds and then start running at line 100. Line 100 or some subsequent line should have a RESUME line no. statement, or the next time an AFTER statement is encountered or an error is trapped, an error will occur. This is a nice feature; it allows true timed interrupts and restarts.

Memory Management

The OPTION BASE command may be used ONLY ONCE in a program and must be used before any arrays are mentioned.

The description in the manual of how one uses the VARPTR function to find the

Continued on 19

INTERACTIVE, USER-FRIENDLY, MENU-DRIVEN
by Donald Forbes - JACG

If you want to make a living as a programmer today, you might do what I just did: post a sign over your console reading: Interactive, user-friendly, menu-driven. Those words hold the key to success. Dr. An Wang grosses a billion a year with his ubiquitous word processors. You never see any source code, just screens with messages telling you which button to push next. The secretaries love him and he reciprocates with bumper stickers: Have you hugged your WANG secretary today? Wang, you know, means King in Chinese.

The user-friendly touch made BASIC what it is today, the favorite of the micro owners. BASIC was developed almost 20 years ago for students (the B stands for beginners) who rebelled at the unfriendliness of FORTRAN. A sympathetic INPUT N or INPUT A$ allowed them to input numbers or text without the convolutions of FORTRAN's format statements, and the kids took to BASIC like ducks to water. Most experienced teachers of computer languages agree that the BASIC which comes with small computers is not a good language for teaching programming. The GOTO statement encourages sloppy code. But friendliness triumphs over good judgment.

In FORTH you can talk to your computer easily enough. If you load your JACG disk # 20 and type CR ." I love my ATARI" and hit return you get the message back. Start with DECIMAL because BASE defaults to HEX. If you type a number, say 3, and hit return, and then type . (a period) and hit return you get the 3 back. You need something more for a two-way conversation. Two words hold the key to friendly interaction between FORTH and the user. The first one, which gets data into the computer, is (naturally enough) KEY. The second, which outputs data, is EMIT. Both deserve a full explanation.

KEY is unique in FORTH because it has a built-in wait. And there lies the secret. As Milton says: "They also serve who only stand and wait." You interact with the restaurant chef through the waiter who waits on you. If you type KEY and hit return you do not get the usual OK. Your ATARI is waiting for you to press another key. Hit A, and you get the OK. What happened to the A? It is now sitting on the stack. To see it type a . (period) and you get 65 OK. This may not look like an A to you, but to the computer which speaks in decimal and ASCII it is an A. Try KEY <ret> A <ret> EMIT and you get the astronaut's message A ok. For W ok try KEY <ret> W <ret> EMIT <ret>. A neater way is CR KEY <ret> A <ret> EMIT CR.

KEY is good for several tricks. If you misplaced your handy pocket reference card and need the ATASCII code of, say, Z then KEY <ret> Z <ret> . will give you the 90 you wanted. You can use KEY to insist on a password as in : PASSWORD-A BEGIN KEY 65 = UNTIL ; Your ATARI loops until it gets an A. You can use the built-in wait in a DO...LOOP as follows : PIG 10 0 DO ." PIG " KEY DROP LOOP ; where your ATARI waits for a key each time through the loop but then discards it. The wait also lets you use KEY to point to items in a menu, depending on the chosen character.

All FORTH terminal output is handled through EMIT, which takes an ASCII character off the stack and sends it to the screen (or printer). Try it with 66 EMIT <ret> which gives you a B, or 253 EMIT <ret> which rings the buzzer, or : PRINTABLES 124 1 DO I EMIT LOOP ; which prints characters or : INVERSES 255 126 DO I EMIT LOOP ; which prints the inverse characters.

Here is an example from Ekkehard Floegel's excellent FORTH ON THE ATARI which uses EXPECT (built around KEY) and TYPE (built around EMIT) to input and output a mailing label. Remove your #20 JACG disk and replace it with a blank formatted disk. You can put this program on screens 1 and 2 and then LOAD them to run the program. Type INPUT and you will be asked for your name and address and state (two characters). Type OUT and you will get a mailing label on the screen. An excellent example of the way KEY and EMIT work together to give you that warm friendly feeling.

Here is another interactive example, also from Floegel. This example uses WORD to accept characters from the input stream and outputs them with TYPE. The following screen converts your ATARI into a cash register. Type CASH and the cash register will appear on the screen. Input your amounts by typing the $ sign followed by a blank followed by a number containing an embedded decimal point; the amounts will cumulate on the screen.

My favorite interactive game so far is on screen 4, which is designed to play the game 20-QUESTIONS. Type 20-QUESTIONS to play, by entering your questions with a final question mark. The game is from Leo J. Scanlon's scholarly FORTH PROGRAMMING based in turn on a David Thornburg article in COMPUTE! for September 1981.

As you have probably guessed by now, the game is a hoax: you will get a yes if your questions end with the letter "E" or "S" or "L" but anything else gives a no. Repeat a question and you always get the same response. This game has all the friendliness of a mule trader or a used car salesman. But try it out on your trusting friends. They will be awed and mystified by your ATARI's infinite wisdom. They will never ever know the truth if you don't tell them!

There are eleven terminal input-output words (four input: KEY EXPECT QUERY WORD; seven output: CR EMIT SPACE SPACES TYPE COUNT -TRAILING) which take a while to cover, so as we pause for station identification remember...

INTERACTIVE USER-FRIENDLY MENU-DRIVEN

Continued on 20

HOW TO WRITE A PROGRAM

By Dale Eisenberger

(Part II)
Reprinted from M.A.C.E.
August, 1982

Most beginning programmers start out going to the keyboard with an idea, sitting down, and coding a program on the fly. For very simple programs this process is adequate, but anything of quality needs definition and design before beginning to code.

Don't be afraid to spend most of your time defining what you want to do in great detail. Until you have these details, you cannot design the structure of the program. And don't be fooled, you are creating an order from the uncountable variables, and syntax rules, and randomness of the Atari universe. Writing a program takes a great deal of effort. Reward yourself early.

In Basic, to speed programs up, the language requires subroutines,(and some programmers feel FOR/NEXT loops and GOTO's should be in the front of the program. The nature of the system says that when the program does any branching, it has to go back to the beginning of the program to find where it left off. This is not good for beginning programmers. It disturbs the logical sequence and flow of the program, for the sake of decreasing the run time of the program. I feel that the most emphasis should be on organizing the program in such a fashion that it is easy to find logic errors, and easy to understand. Once the program is working correctly, you can begin to rearrange it to make it run faster. Yes, it is harder to retrofit time-saving code, but you will then only have to test the speeding up of the code, not the work of the program itself.

When you look at a double-layer, double chocolate cake, your mind often wishes it could engulf the entire cake with a single gulp. Speaking from past experience, let me assure you IT WON'T WORK! You have to cut it down to size. Cut the cake into sections for each person at the table. Some people want a large slice, some a small slice. In the same way, you must divide your program into sections. A section of code to set switches and values will probably be smaller than a section that processes a particular type of record. Just as a whole cake is too much to eat in one byte (excuse me, bite), a piece of cake is also too much. We have to take a fork to cut it down to a size we can eat. Your section of code has to be broken down to a size where you can know all of the variables that affect this section of code, what this section of code will do to these variables, and where in the program will the computer go next.

TO SUMMARIZE:

1. Spend most of your time defining and designing what you want your program to do in great detail.
2. Getting the program working correctly comes first; worrying about the time it takes to run the program comes later.
3. Cut you program down into the smallest logical sections, until you understand everything about the section. ▲

PROGRAMMING WITH STYLE

by John Carmody

Reprinted from Houston A.C.E.
May, 1982

This column will talk about internal style in programming. Two levels of programming must be differentiated before we can proceed: When the program (plus data in memory) is large and little space is left, or extreme speed is required; when size and speed are not overly critical. In the former case the program should be written to be the shortest or fastest possible. In the later case the program should be written in the clearest, most logical, straightforward manner possible. It is this latter case I am going to discuss, since most of us are in this situation.

The easiest way to get a clear, logical program is to write it as a series of modules. This makes writing later programs easier too, since modules are often useable in many programs, if well designed. So how do you design a module? One way is to decide what modules you will need, then decide which need to be general so that they can be used in other programs. Write the general module first. Decide what variable names you will use in these modules then use them appropiately in the rest of the program so that they have the right values when you enter the module. The module will require some variables for use as counters, etc. be careful of these. For example if you are in a FOR-NEXT loop using I as the index and GOSUB the module which also uses I as an index for a FOR-NEXT loop, when you RETURN I will no longer have the correct value (unless the module explicitly resets it). A useful procedure is to note down each variable as you first use it, including its expected use and range. In transportable modules the programmer should attempt to avoid using GOTO and GOSUB. Where these are unavoidable- use variables as arguments, not numbers. This is to make it easy to renumber the module if required. Each module should be independently error protected, although each may TRAP to an error handling module. If many modules are to refer to an error handling module it must be completely general and should allow the user to choose an appropriate action before

doing anything drastic, but should recover from errors by itself whenever possible---don't forget to inform the user of the error though. Modules should NEVER GOTO outside the module. The correct way to use the modules is to have a main program which calls the main modules, which call specific modules, etc. Breaking this rule will be very difficult to modify and almost impossible to debug. Having written some of the modules, and thought about the rest it is now time to write the control or main program. This is just a series of calls to modules, with some decision structures. Its function is to solve the original problem using the modules.

At this point you will find that your concept of what the modules should be will have modified, and you will modify the modules. After debugging you will finally have a running program. You are through...WRONG!!!!

Before you are through with any piece of softwear you must DOCUMENT the program. There are two types of documentation, internal and external.

Internal documentation includes the use of meaningful, variable and module names (GOSUB TOUCHDOWN rather than GOSUB I), and REM statements explaining the program. The REM statements should be used liberally throughout the program. If possible explain virtually every line. Use more documentation to explain module function than for smaller structures.

Internal documentation is for the programmer's use, yourself included. You will forget the details of a program in a couple of months, and when it needs modifying you will be as lost as anyone else without documentation. External documentation can take many forms, but the two most common are aimed at programmers and users. Documentation for programmers is similar to REM statements, but both more general and more specific since space is not at a premium. Documentation for users should explain how to use the program, its functions, errors that can occur, any input and output required, etc. ▼

THE TOOLSHED
by Pat Warnshuis
PAC Newsletter, January, 1983
Retyped by Howard Johnson

Wynn Smith, of Mosaic Electronics, sent me a card with a really fun idea. Don't et started with this thing until you've said goodnight to your loved ones. In fact, better not get involved on a work night. OK, everybody to the keyboard!

Start off nice and easy: in the direct mode, type POKE 53261,128:POKE 53248,60. See that nice, fine line down the left side of the screen? That's Player0 from the player-missile family (one of the original Coneheads). So what? Well, that line would look the same in any graphics mode. For Graphics 0, you could use it to set off a column of numbers. 53261 sets the shape for the Player0 line. Just POKE 53262 with a number for a line using Player1, POKE 53263 for Player2, etc. 53248 controls where the line will appear on the screen. On my TV, line 50 is the left margin and line 208 is the right margin. Try it to find yours.

Now, change the line by POKE 53261,129. This gives you two lines. You could move them to bracket a 2-number column of figures. Remember, you can add more lines using the other Players in location 53261 to 53265. You can move the lines around using locations 53248 to 53254.

Now the big stuff! POKE 53261,255. A nice broad stripe, right? POKE 53256,3 to make the line 8-columns wide, or POKE 53256,1 for a 4-column stripe. POKE 704 to 707 to change the colors of the stripes for Player0 through Player3. Think how you could use this for highlighting columns in a display screen. Or POKE 53256,2 to get back to normal width and see how you could add colored borders on the sides of your screen.

Hey, try this:

```
100 POKE 623,32+16+1


101 REM combines the four 2-column missiles into one player, color in 711


110 POKE 53261,255:POKE 53248,52:POKE 704,16*3+2:POKE 53256,3


120 POKE 53262,255:POKE 53249,84:POKE 705,16*5+4:POKE 53257,3


130 POKE 53263,255:POKE 53250,116:POKE 706,16*14+6:POKE 53258,3


140 POKE 53264,255:POKE 53251,116:POKE 707,16*2+8:POKE 53259,3


145 POKE 53265,255:POKE 53253,170:POKE 711,16*2+4:POKE 53260,3
```

Now, in the direct mode again, try:

```
FOR H=50 TO 206:POKE 53248,H:FOR I=1 TO 10:NEXT I:NEXT H
```

Notice the effect of the overlayed colors as one stripe passes over the others. Dink around with it. Change width, color, shape and position. Remember, it works in any graphics mode so you get free vertical lines or area highlighting in additional colors. Meanwhile it should desensitize you to using PM graphics. Check your Memory Map later to see what you're controlling. I like it! ●

BOOLEANS

By Chris Fostel

Reprinted from P.A.C.E.
December, 1982

The word BOOLEAN congers up images of something more dreadful than a Battlestar in STAR RAIDERS or an Umberhulk in ROGUE. Is it one of those grisly secrets that you should know to successsfully navigate the strange world of computers, but hope you can survive without? Actually, the term refers to a system of logic developed by the ninteenth century mathematician George Boolean. [sic] Boolean was interested in finding a mathematical way to solve problems in logic - a field previously untouched by math. Assuming that there are only two states in pure logic, TRUE and FALSE, he defined a set of "operators" <,>,<=,>=, etc. similar to the familiar algebraic operators (+,-,/,*), and a set of rules for their use.

When you think about the close relationship between TRUE/FALSE and ON/OFF, you realize that had George Boolean owned a digital computer he would have been the father of today's programming. As a result Boolean logic has many applications in today's programming. Unfortunatly, the ATARI BASIC manual only mentions that Booleans are available in ATARI BASIC and then goes on to another topic. How much of George Boolean's creation is available? How do you use it?

While I am certainly no expert at Boolean logic it seems ATARI went far beyond merely using the Boolean operators when the ATARI BASIC was designed. I was delighted to run across three examples of how to use Boolean's in more creative ways. The first was actually some time ago when David and Sandy Small wrote the OUTPOST:ATARI articles for *CREATIVE COMPUTING*. It concerns the use of "flags" - little half variables generally with values of either 1 or 0 that can be used along with branching statements to change which path a program will take if some condition becomes true. For example, suppose you want to give some instructions on how to use a subroutine on the first pass through, but don't want to see the same instructions on the second or third pass. It can be done like this:

```
180 IF FLAG=1 THEN 400
190 FLAG=1
200 REM START OF INSTRUCTIONS
   e
   t
   c
   .
400 REM BEGINNING OF SUBROUTINE
```

Simple enough? But ATARI'S use of Booleans allows line 180 to be shortened to:

```
180 IF FLAG THEN 400
```

The program will take the branch to 400 if the value of FLAG is anything except zero! It may seem trivial, but it saves key strokes, about 8 bytes of memory and it allows FLAG to be used as a full variable (as long as it never again is 0). With a long program or many flags the savings could add up.

The second example deals with that old pain, reading the joystick. The joystick can have 9 values. To read them all most programs have eight IF-THEN statements:

```
100 IF STICK(0)=14 THEN Y=Y+1
110 IF STICK(0)=10 THEN X=X-1:Y=Y+1
120 IF STICK(0)=6 THEN X=X+1:Y=Y+1
130 IF STICK(0)=13 THEN Y=Y-1
140 IF STICK(0)=11 THEN X=X-1
150 IF STICK(0)=9 THEN X=X-1:Y=Y-1
160 IF STICK(0)=5 THEN X=X+1:Y=Y-1
170 IF STICK(0)=7 THEN X=X+1
```

BORING? YES. NECESSARY? NO!

There have been many articles on how to read the joystick more easily than using the above eight IF-THEN's but none of them quite meets the beauty of using Booleans. When the computer encounters an IF-THEN choice it creates an internal flag within the hardware. If the statement is TRUE the internal flag will have a value of 1, if the statement is FALSE then the internal flag will have a value of 0. Atari BASIC lets you use this internal flag directly in algebraic/boolean equations. The joystick reading problem can be reduced to three lines of code. I'm going to use 5, because I don't want the lines to spill over when printed in 40 columns.

```
100 Z=STICK(0)
110 X=X+((Z=6)+(Z=7)+(Z=5))
120 X=X-((Z=11)+(Z-10)+Z=9))
130 Y=Y+((Z=10)+(Z=14)+(Z=6))
140 Y=Y-((Z=9)+(Z=13)+(Z=5))
```

(Lines 110 and 120 could be combined, so could lines 130 and 140 to make this a three line joystick reading routine.)
What happens is, Z (the joystick) will have only one value - let's say 14. When the program executes line 110 it will check Z against all of the values in the individual comparisons. If a comparison is TRUE it will be replaced by a value of 1, if False it will be replaced by a value of 0.
Therefore, assuming Z=14 when executed the joystick routine becomes:

```
110 X=X+((0)+(0)+(0)) :REM Z=14
120 X=X-((0)+(0)+(0)) :REM Z=14
130 Y=Y+((0)+(1)+(0)) :REM Z=14
140 Y=Y-((0)+(0)+(0)) :REM Z=14
```

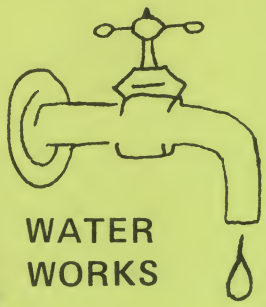
This same result the earlier IF-THEN statements would have gotten in nauseating detail!!!!

Continued on 18

GORILLA
12" GREEN
94.95

AMDEK
COLOR 1+
$299.95

OR MAZE



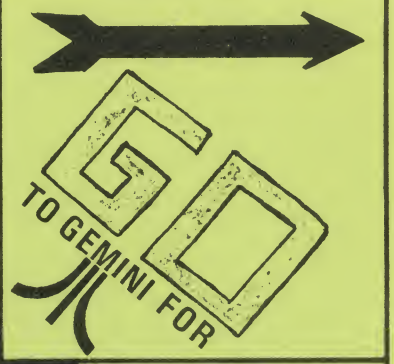
RANA 1000
Single, Dual, Double
$299.95

ATARI 1050
Single, Dual
$339.95

TRAK AT-D2
Single, Dual, Double
with Printer Port
$379.95

PERCOM AT-88
Single, Double
with Printer Port
$ CALL $

DISK DRIVE



OUR ATARI® CONNECTION!

EMINI HAS A

OPOLY

FORMANCE AND SERVICE

DON'T TAKE THE CHANCE • SHOP GEMINI FOR QUALITY & VALUE CHANCE

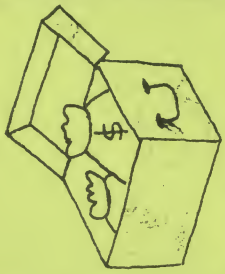
DALE AVENUE, CEDAR KNOLLS, N. J.
FRI 9-8 • SAT 9-4 •CLOSED SUNDAY

INTERFACE ROW

ATARI 850
$164.95

MICROBITS
PARALELL PRINTER
$84.95
Complete with Cables

ATARI CHEST



Follow Instruction On Top Card

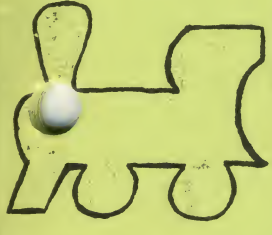
PROCESSING PLACE

40/80 COLUMN
LETTER OR DATA PERFECT
$84.95 EA.

ATARIWRITER
$79.95

TEXT & SPELL WIZARD
COMBO $64.95

ATARI RAILROAD



PRINTER PROMENADE

ATARI
1020 $239.95
Color Printer
1025 $399.95
Dot Matrix
1027 $269.95
Letter Quality

EPSON
RX 80 $299.95
RX 80F/T 449.95
FX 80 549.95
FX 100 689.95

C. ITOH
DOT MATRIX
PROWRITER
$369.95

GEMINI
DOT MATRIX
10X $319.95

SILVER REED
500 449.95
550 649.95
Letter Quality
Daisy Wheel

PATCHING UP BASIC A+

By Joseph M. Apice - JACG

If you have purchased BASIC A+ from O.S.S.,you have discovered what real power you can have on you ATARI 800 when given the right basic.

One nice feature of BASIC A+ is that it is disk based and will allow user installed patches. Another is that it looks like a cartridge based system when used with the standard ATARI DOS. This means that it does not try to reboot or access DOS every time the RESET key is pressed. It simply clears the screen and returns the READY prompt,(well almost!).

This brings us to the heart of the article. BASIC A+ with all its nice features still utilizes the ATARI default screen comprised of a large blue square surrounded by a block border. In addition we are greeted by a header at the top of the screen identifying the program version and the manufacturer.

If there's one thing that I can't stand, it's headers popping up every time the RESET key is pressed. We all know who markets it. It's on the disk label,it's in the manual, and it's even in the magazine advertisements, so why must we be constantly reminded of it on our screen. As for ATARI! I still don't know why they chose that big blue square and black border. I always feel that I'm in a maze and PACKMAN is about to appear and eat my READY prompt.

What were we talking about?. Oh yes!,BASIC A+. Well, if you would like to add a little appeal to your screen and at the same time remove that header try the following patch. The patch will give you a full screen in royal blue(no border) and at the same time eliminate the header. Use the Assembly Language cartridge or equivalent to generate the code and please save the new version on a disk other than the original BASIC A+ disk in case you make an error.

```
 10  *=$8549
 20  LDA #146
 30  STA $02C6
 40  STA $02C8
 50  STA $0052
 60  NOP
 70  NOP
 80  NOP
 90  NOP
100  NOP
110  NOP
120  NOP
130 END
```

Save the code using ASM ,,#D1:PATCH.OBJ on your auxiliary disk.

Now boot the BASIC A+ disk and load BASIC.OBJ using the L option from DOS. When the READY prompt appears type DOS. Insert the disk containing the PATCH.OBJ and use the L option again to load the file. DOS will load PATCH.OBJ and return with the prompt "SELECT ITEM OR RETURN FOR MENU". Select option K(Binary Save) and save the customized version of BASIC A+ as NEWBASIC.OBJ,8400,BB00,8400 or some equivalent name.

You now have a modified version of BASIC A+ without that traditional ATARI look. You will find the patch especially useful when using a monochromatic monitor. Use 0 in place of 146 [at line 20] for green phosphorous monitors to provide a cleaner background and that professional look.

Other patches are available using the DEMO disk available from O.S.S.. The disk contains several utilities including XREF,a variable cross reference utility,RENUM,for renumbering program lines,LOAD @ which loads ATARI programs written via cartridge and M:driver which allows memory to be used as a device. There are also examples of player missile programs,games and other nice goodies.

The disk is available free of charge from O.S.S. However, quantities are limited and if you have problems getting a hold of one, please notify one of the J.A.C.G. officers and I will make a copy available through the user library.

I would like to thank Bill Wilkinson of Optimized Systems Software for his help in locating the header addresses which allowed me to make this patch.

JACG MEMBERSHIP

The Jersey Atari Computer Group (JACG) invites you to become a member. Dues are $20.00 per year and entitle the member to 1) Receive the monthly newsletter; 2) Purchase programs from the group's extensive tape and disk librarys at special rates; 3) Join special interest groups or form new ones; 4) Benefit from the expertise and experience of other Atari computer users; 5) Participate in group purchases of software at substantially reduced prices; 6) Receive a membership card that entitles the member to discounts at local computer stores; 7) Attend monthly meetings to learn about the latest hardware and software, rumors, and techniques for getting the most out of your Atari computer; 8) Submit articles and programs to the newsletter and give demos and presentations at the monthly meetings; 9) Participate in sale/swap activities with other members; and 10) Have a lot of fun.

If all of this sounds good then send a check or money order, payable to JACG, to:

Rick Olson
5 Starling Drive
Randolph, NJ 07869

Remember, receiving the JACG Newsletter is just one of the many benefits of being a member of JACG.

TRANSLATING OTHER BASICS INTO ATARI BASIC

by William Frank
Reprinted from H.A.C.E

1. DEF(X)=INT(RND(0)*7.98+1.01)
   ATARI has no DEF so you either have to define the function each time it appears or use a GOSUB.

2. IF...THEN...ELSE
   ATARI has no ELSE so you need to make the ELSE part of the next line.

3. IF...THEN
   In ATARI Basic if the condition is not met, control passes to the next line. Use multiple statements on the same line # as an IF statement only if they are also conditional.

4. LEFT$(A$,Y) returns the Y leftmost characters of A$
   ATARI has no LEFT$ so you need to use A$(1,Y)

5. MID$(A$,Y) returns the Y characters of A$ starting with the X character.
   ATARI has no MID$ so you need to use A$(X,X+Y-1).

6. RIGHT$(A$,Y) returns the Y rightmost characters of A$
   ATARI has no RIGHT$ so you need to use A$(LEN(A$)-(Y-1),LEN(A$)).

7. A$(I) means I strings
   ATARI has no string arrays. Use A$(I*(B-1),I*B) where B is the length of the longest string.

8. A$=A$+B$
   A$(LEN(A$)+1)=B$

9. CLS means clear the screen
   GR.0

10. TEXT means enter Text (display) mode
    GR.0

11. HOME positions cursor on APPLE
    POSITION 2,0

12. PRINT TAB(12);"HELLO" means print HELLO 12 spaces over
    ATARI has no TAB function, so use
    POSITION PEEK(84),PEEK(85)+12 where
    PEEK(84) is the current row and
    PEEK(85) is the current column

13. VTAB 12
    ATARI has no VTAB. Use POSITION PEEK(84)+12,PEEK(85)

14. HLIN A,B AT C Apple Horizontal Line.
    PLOT A,C:DRAWTO B,C

15. VLIN A,B AT C Apple Vertical Line.
    PLOT C,A:DRAWTO C,B

16. SET A,B (Apple)
    PLOT A,B

17. RESET A,B (Apple)
    COLOR 0:PLOT A,B

18. PRINT @ 234 TRS-80 Position Statement
    Use POSITION STATEMENT with LINE=POS/40, COL=POS-(LINE*40)

19. PRINT USING$###.##;X TRS-80 format allowing only two decimal places.
    X=(INT(X*100))/100:PRINT"$";X

20. NUMBER# # means double precision (12 place instead of 6 on TRS-80)
    ATARI has no equivalent.

21. INPUT "How many turns";X
    PRINT "How many turns";:INPUT X

22. INPUT A(3)
    INPUT Z:A(3)=Z

23. RND(3) On some computers a random number of 0 to 3.
    RND(0)*3 ATARI only gives zero to one.

24. ON ERROR GOTO 1000
    TRAP 1000

As in all life nothing is totally free. The ATARI is far superior to other computers in having three microprocessors instead of one. This is what allows for the fantastic sound and graphics. To allow for the sounds and graphics and not have BASIC take up more than 8K, some minor sacrifices were made. The above shows that there are not to difficult ways around all of the omissions from the ATARI BASIC. I for one am convinced we have gained far more that we have lost!!!

▼

INTRODUCTION TO COMPUSERVE SIG*ATARI

By John R. Babson
Re-typed by Howard Johnson

The SIG*ATARI section of COMPUSERVE is found by entering "GO PCS 132" at the prompt "!" at the end of the menu. Just to get you started and hopefully to save you some time and money let me suggest the following procedure for your first try in this section. After you GO PCS-132, you will see something like the following:

```
Welcome to SIG*ATARI, V.1A(45)
Name:    JOHN DOE 71000,123
Last on: 18-Dec-82 22:52:53
High msg#: 3515
You are user number 6551
System contains messages
3277 to 3627
Enter blank line for menu:
SIG*ATARI
Function menu:
1 (L) Leave a message
2 (R) Read messages
3 (RN) Read new messages
5 (B) Read bulletins
6 (CO) Online conference
9 (OP) Change your SIG options
0 (E) Exit from this SIG
Enter selection of H for help:
```

At this point enter:OP

Continued

Booleans

The final example is another familear problem - recognizing that the cursor has gone beyond the limits of the screen and fixing the values before they cause an ERROR code to stop execution. You could use:

```
200 IF X<0 then X=0
210 IF X>LIM THEN X=LIM
```

Booleans allow this to be done in a single statement:

```
200 IF X<0 or X>LIM THEN X=LIM*(X>LIM)
```

If X>LIM is TRUE then the boolean will have a value of 1 and X will get the value of LIM. However,if X>LIM is FALSE (the other possible value from the "OR" is X<0 then the boolean will be 0 and X will get the value of zero! It's not as clear that this is faster than the original two IF-THEN's but it certainly looks nicer.

Enjoy these "new" programming hints and let me know if you find other uses for ATARI booleans.
(P.S. Ask an Apple, Trs-80 or C/PM user if they can use booleans in any of these ways.) ■

Writing for the JACG Newsletter

The JACG Newsletter seeks hardware and software reviews, tutorial articles, programs and any other information of interest to Atari computer users. Material should be sent to the Editor (see back page of this Newsletter for address) and conform to one of the following formats:

1) LJK Letter Perfect files on disk, 2) Text Wizard files on disk, 3) *4-1/4* inch column, single-spaced, dark black ink, right justified, no printing on perforation and pica font (10 cpi) hard copy, 4) AtariWriter files on disk, 5) Bank Street Writer files on disk, 5) BASIC REM statements on disk, and 6) BASIC REM statements on tape. Anything else will not be accepted, especially hand written or non-justified type written. Figures should be in black ink and camera ready. Programs should be submitted on disk accompanied by a listing. If the program does not work as indicated, it will be returned.

The above format options are numbered from 1 through 6. The *lower* the number of the option you choose in submitting items for the JACG Newsletter, the *better* the Editor will like you and the *sooner* the item will appear in the Newsletter.

The Editor reserves the right to make changes, accept or reject submitted material. ■

Compuserve

This command 'OP' takes you to the menu for setting various options. If you set these before reading or scanning messages it can save you some time. The menu that will appear will be the following:

```
User Options menu:

1      Change to command mode
2 (LL) Change line length
3 (T) Return to Function menu
0 (P) Make options permanent
Enter selection or H for help:
```

At this point you should select option 1 and then you will be asked to identify the user options you wish to select. The ones to use are:
NS - don't stop between messages
BR - set brief mode, which suppresses repetitions display of options
T - return to Function level

The sequence will appear as follows:

```
Enter selection of H for help:1
User option:BR
User option:NS
User option:T
Function:
```

At this point give the function command of 'QS' for Quick Scan. This command will scan the message files and identify the message number and the subject of the messages you are interested in. subject of the messages you are interested in. Fortunately this command does not identify every message that is on the system. It shows only the first message on a particular subject and identifies how many replies to that message oar on the system. It is then possible to use the 'RT' command to read the main message and all replies to it. After you have scanned using the QS command and have identified the message numbers you are interested in, use the following command:

```
RT #### ONLY
```

where #### is the number of the message you want to read.

After this command the message you requested and its replies will be displayed without stopping between messages and without displaying the various rerepetitive commands. ■

location of a string is a bit confusing. The function VARPTR(A$) returns a value; this value is an address. At this address is stored the current length of the string A$. The two bytes following this address contain the address (Lo byte, Hi byte) of the string A$ itself. These statements may sound confusing but I believe that they are correct. Read them carefully and I think that it will become clearer.

Input

Input statements are somewhat different from 8K BASIC. First of all, a null entry (just typing RETURN) causes a variable to retain the original value; a string is not nulled and no error is generated if the input value is numeric. If you attempt to input a non-numeric value to a numeric variable, the error is automatically trapped with the prompt ?REDO FROM THE START, and the program returns to the input statement in which the error occurred.

Subscripted variables, both string and numeric, may now appear in INPUT statements; however, there seems to be a slight bug that can turn up on occasion. If you are using a FOR ... NEXT loop to input a string array, e.g.

```
FOR I = 1 to 10:INPUT ST$(I):NEXT
```

the first character of the first string, ST$(1), can get lost. This can be prevented by printing each string immediately after it is input; this seems to "set the hook", so to speak. This may not always occur, but be aware of the possibility.

One last point about INPUT, there is no such command as INPUT ... AT (or GET ... AT for that matter). I don't care what the manual says, this set of commands is not implemented, at least not in the version that I have. If you try to use these commands, whether from the screen or a disk file, you get a SYNTAX ERROR every time.

Don't get too upset about this; we still have random-access disk files. Even though the POINT command of 8K BASIC is gone, there is an equivalent in the PRINT ... AT; command of Microsoft BASIC. This command without an output list acts as a cursor positioner on the screen or the sector-byte pointer in a disk file. The disk file may be opened for input only as long as the PRINT ... AT; command has not output list (i.e. PRINT #2,AT(123,45); not PRINT #2,AT(123,45);X;Y;Z. After the cursor or file pointer has been positioned with this command, a simple INPUT or GET command will read in the data. Oh, yes; INPUT doesn't work from the screen at all, you must use a GET#6 or a GET #0 to read data from the screen. Yes, that's GET#0;IOCB#0 is completely accessible in Mirosoft BASIC. Be careful, though; a simple CLOSE#0 entered from the keyboard will cut off all communication with the system and you will need a SYSTEM RESET to reestablish it.

Odds and Ends

In defining functions, the variables that appear in the function name are locally defined and are free to be used elsewhere in the program. That is, in the definition

```
DEF FUN(X,Y)=X*Y+B
```

X and Y are locally defined but B is a global variable. Furthermore, function definitions must be entered as program lines and RUN, i.e., program control must pass over the definition lines, before the functions can be used. From this time on, as long as the program is not altered or edited in any way, the functions are accessible both to the program that is running and from the immediate mode as well.

The reason that there must be no editing of the program in order to keep use of the functions is that Microsoft BASIC resets everything if any program editing is done. That even includes entering a blank program line that doesn't change anything in the program. This has the unfortunate effect that the ability of the Atari operating system forced-read mode to allow a program to change itself while it is running is rendered practically useless.

VAL("HELLO") returns a value of zero rather than an error as in 8K BASIC. This is useful but could cause trouble if you are not aware of it.

PLOT TO X,Y may be used alone, and functions exactly like the DRAWTO X,Y command of 8K BASIC.

Watch out for the AUTO command; if no beginning line number is given it starts at the last line edited plus the increment, not the highest line number plus the increment. It operates very much like the INPUT statement with respect to prompts. The line number that is printed on the screen is a prompt; the line number has already been entered into memory, and the prompt is ignored as long as the cursor is kept within the logical line. If, however, the cursor is moved to another line, or just out and back into the original line, the line number on the screen will be read in as part of the text of the original line. For example, AUTO prints 100 on the screen; if you move the cursor out and then back into the line and then type REM, the LIST command will show that you have

```
100 100 REM
```

and this will cause trouble.

The trace function initiated by TRON (the command, not the movie) can get fouled up if the program it is tracing is also doing a lot of writing to the screen.

Continued on 20

The two outputs seem to get tangled up with each other.

MERGE works on SAVE'd files just as well as on LIST'd ones. I know the book says no, but in my system it works.

How do you get a number into the variable A in the machine language routine called by A=USR(addr,PARM)? When you jump to the routine, it wll find the two byte number PARM in locations $E3 and $E4 in Lo byte, Hi byte form, and in locations $E9 and $EA in Hi byte, Lo byte form. This much the manual tells you, albeit in a rather cryptic form. What they didn't tell you is that when the routine does an RTS to BASIC the number left in $E9,$EA in Lo byte, Hi byte form wll be transferred to the variable A as a signed two-byte integer.

Bugs?

LOG(1) yields a value of 2,32396e-08, not the value given in the book.

If you use the command AFTER (time_in_jiffies)line_number, a subsequent RENUM command will NOT RECOGNIZE the line number. If you use the optional form AFTER (time_in_jiffies) GOTO line_number, everything works okay.

I have heard of one bug concerning the down-arrow key not working properly with large programs in memory. I have never had this experience; perhaps I haven't had big enough programs, or perhaps this bug is apocryphal, but keep it in mind.

eXIt

Atari Microsoft BASIC is a powerful, high-level language. In many ways,it is better than you would think just reading the manual. It has most of what one would want in a programming language. There is just one thing I really want; I miss the XIO commands.

P.S. I would appreciate hearing about any errors, typos, or other relevant information you have to offer. Send to:

Frank C. Jones
416 Hillsboro Drive
Silver Spring, MD 20902
(301)593-1056(h)
(301)344-5506(o)

FORTH

```
SCR #  1
00 ( interactive floegel )
01 : FNAME PAD 10 EXPECT ;
02 : LNAME PAD 10 + 10 EXPECT ;
03 : STREET PAD 20 + 15 EXPECT ;
04 : CITY PAD 35 + 15 EXPECT ;
05 : STATE PAD 50 + 2 EXPECT ;
06 : ZIP PAD 52 + 5 EXPECT ;
07 : INPUT 125 EMIT CR PAD 58 32
08 FILL ." FIRST NAME " FNAME CR
09 ." LAST  NAME " LNAME CR
0A ."      STREET " STREET CR
0B ."        CITY " CITY   CR
0C ."       STATE " STATE  CR
0D ."         ZIP " ZIP    CR ; -->
0E
```

```
1E
1F
SCR #  2
00 ( interactive floegel 2 )
01 : TYPE -DUP IF OVER + SWAP
02 DO I C@ 127 AND DUP 0=
03 IF DROP ELSE EMIT THEN
04 LOOP ELSE DROP ENDIF ;
05
06 : PRINT PAD + SWAP -TRAILING
07 TYPE ;
08 : OUT CR 10 0 PRINT SPACE
09   10 10 PRINT CR 15 20 PRINT
0A CR 15 35 PRINT SPACE
0B 2 50 PRINT SPACE 5 52 PRINT
0C CR ;
0D
```

```
SCR #  3
00 ( cash register floegel p 73 )
01 DECIMAL
02 : CURS 85 ! 84 C! ;
03 : >$ <# # # # 46 HOLD #S 36 HOLD
04   #> ;
05 : .$ OVER OVER >$ DUP 34 SWAP -
06   1 SWAP CURS TYPE ;
07 : CLR 5 2 CURS 20 0 DO 32 EMIT
08   LOOP 4 2 CURS ;
09 : $ 13 WORD HERE NUMBER D+ .$
10   CLR QUIT ;
11 : CASH 125 EMIT 1 2 CURS
12   ." CASH: " 0. .$ 4 2 CURS
13   ." INPUT:" QUIT ;
14
```

```
31
SCR #  4
00 ( scanlon 20-questions) DECIMAL
01 : LENGTH 255 0 DO DUP I + C@ 0=
02 IF I LEAVE THEN LOOP SWAP DROP ;
03  : PXD 30100 ; ( memory locatn)
04 : 20-QUESTIONS 125 EMIT
05   ." What is it ?????? "
06   21 1 DO CR CR
07   ." enter question  " I . CR
08 PXD 60 EXPECT PXD LENGTH
09 PXD + 2 - C@ CR DUP 69 =
10 IF ." Yes!" ELSE DUP 76 =
11 IF ." Yes!" ELSE 83 =
12 IF ." Yes!" ELSE ." No"  THEN
13 THEN THEN LOOP CR
14 ." That's twenty questions!" CR
15 ." Type 20-QUESTIONS to play" ;
16
```

integer, real, array, amd string types. This is a powerful tool in the hands of an advanced programmer. All variables must be declared as being one of the data types, and must be given a SCOPE of global (valid within the entire program) or local (within a given procedure). In BASIC, vaiables are always global.

Control statements used are REPEAT .... UNTIL ; WHILE... DO ; FOR ... TO . Conditional statements used are IF... THEN ; IF ... THEN ... ELSE ; GOTO is seldom used, but is valid when line numbers are declared using a LABEL declaration in the form LABEL 100.

Another useful statement is CASE. It allows the selection among several different alternatives based upon the integer value of an expression. The form is:

```
CASE ......... of
  1. statement
  2. statement
  3. statement
```

The general format of a Pascal program is:

```
PROGRAM name (Input,Output);
   definitions and declarations
BEGIN (*program name*)
   body of program
END. (* program name *)
   data
```

When used correctly as a structured program, the main body calls a procedure, which is one of the previously mentioned paragraphs or blocks. Each procedure BEGINS and ENDS, after which control returns to the main program. Closely related to a procedure is a Function. A procedure is a set of actions, while a function calculates a specific value, and returns that value to the main program.

Pascal also handles Arrays, Files, Records, and Sets.Here is an example of a program written in Pascal which reverses 2 integers:

```
Program Reverse (input,output);
  (* reverses two input integers *)
const PrintSpace=1;
var First,Second: integer;

Procedure GetTheNumbers;
     begin
       writeln ('This program reverses two
integers.');
          writeln ('What is the first
number?');
       readln (First);
          writeln ('What is the second
number?');
       readln (second);
     end; (* GetThe Numbers *)

Procedure SwitchThem;
   var Temporary:Integer;

     begin
       Temporary:= First;
       First:= Second;
       Second:= Temporary;
     end; (* SwitchThem *)

Procedure PrinttheResults

     begin
       write ('In reversed order, the two
numbers are ');
       writeln (First:PrintSpace, ' and ',
Second:PrintSpace, '.');
     end; (* PrintTheResults *)

BEGIN (* The main program- Reverse *)
    GetTheNumbers;
    SwitchThem;
    PrintTheResults;
END. (* of the program reverse *)
```

Analysis: As you can see, this is quite a bit different from BASIC. First there are no line numbers, although the program will probably be entered into an editor that has line numbering. Control goes to BEGINing of the main program which calls 3 different subprograms.

First and Second are Global variables which are valid for the entire scope of the program, but Temporary is only valid within the procedure SwitchThem. You may write in upper or lower case, and in a free unstructured format that aids readibility.

The program is specified and involving both input and output, and the variables must be declared as being either Integer, Real, CHARacter or Boolean.

This is quite different from BASIC- it is more time consuming and requires you to concentrate. However it provides a lot of flexibility, and forces the programmer to think structured. It is for this reason that Pascal is a favored language in institutions teaching advanced programming theory.

This completes a brief look at Pascal, and should raise more questions than the answers that I have provided. Next month we will take a brief look at Assembler. ■

JACG NEWS
NEWSLETTER
VOL. 2 NO.1
SEPT. 1982
THE JERSEY ATARI COMPUTER GROUP

PILOT

by Curtis Springstead - JACG

Originally created as an educational authoring language, PILOT (Programmed In quiry, Learning or Teaching), has been prsented by ATARI for this purpose as well as an easy to learn programming language useful in teaching computing. Dr. John Starkweather first created PILOT in the mid sixties in an effort to create a tool for educators who were then just beginning to experiment with computer aided learning (CAI).

After some work with the language Dr. Dean Brown of the Stanford Research Institute Education Laboratory began experimenting with the use of PILOT as a tool for teaching programming. From 1967 to 1974 the PILOT language was exposed to a wide audience of educators who found it most useful but lacked hardware and support in the schools for programming. While less than a decade ago this was still the precomputer era as far as the home and school were concerned.

With the rapid deployment of computers in the home and school the need for easy to learn languages gained new meaning beyond any level experienced before. In answer to this need ATARI released PILOT for their new 400 and 800 models. As a further enhancement, TURTLE GRAPHICS were added to the package. This addition served two purposes, it allowed even beginners to play with the graphics of the ATARI systems and it also made the PILOT package more competitive with another language which was quickly gaining in popularity, LOGO.

So what is so special about PILOT? To begin with it has only 10 or so basic commands to be learned. They represent the basic structures or icons necessary to describe the fundamental functions performed by a computer program. The second feature, strong string handling functions, were a direct result of the intended use of the language in authoring CAI material. Despite claims by other writers, PILOT provides excellent training material for the fledgling computer programmer. The use of modules in the programs developes good structured modular programming which has been demonstrated to contibute to smooth performing programs that are easy to maintain and change.

Here is an example of a basic function you will find in most programs, that is asking the user if they need help and then asking them to respond either yes or no.

```
T: DO YOU NEED FURTHER INSTRUCTIONS ?
A:
M:YES,YEP,Y,OK,SURE
UY:*INSTRUCTIONS
JN:*BEGIN
```

The most frequent problem with this type of request is that even if the user is given prompts for the acceptable values to be entered, some people don't read or pay attention to them. This program simply TYPES the question on the screen and then waits to ACCEPT it from the keyboard. When return is pressed the value ACCEPTED is MATCHED to the list of choices that I have decided to allow for a positive response. If a match is found, indicated by the 'Y' qualifier on the USE command then the module or subroutine INSTRUCTIONS will be executed. If there is no match, the 'N' on the JUMP command, then processing will JUMP or go to the TAG called BEGIN. To be really rigirous in this test a second set of tests could be added to look for acceptable negative responses and failing that return to the operator for another response.

With the instructions given in the sample program you can write many extensive programs in PILOT. The remaining commands add sound, graphics, mathematics and further string functions.

Turtle graphics employ a visible "turtle" which the programmer teaches to walk around on the display to create designs and pictures by turning to angles specified and moving given distances in that direction. The "turtle" can also draw with any of four colored pens. Generally this turtle graphics, first developed by Seymour Papert and a group of computer scientists at MIT, is used to introduce beginning programmers to the idea of sequence and the use of subroutines and modules in programming. They learn that in order to get the shape desired they must give directions very specifically and in the proper sequence. In addition they also learn that if a part of the design is to be repeated it can be "modularized" and requested when needed without being typed in again.

With the announcement of LOGO for the ATARI this fall many believe that PILOT is due for extinction. While it certainly will slow in growth, PILOT is still easier to learn than LOGO for many people and still is preferable as an authoring tool. Beside rumors are about that ATARI is working on a SUPER PILOT that will have many of the advanced turtle graphics features found in the new LOGO. I'm not selling my PILOT cartridge because there are many functions that this language best suits.

●

TERMINAL

Although I have never been very good at the game, I have always liked Q-Bert. Its graphics are the prettiest to be found anywhere. The game is quite challenging too. Who would have thought that a game consisting of hopping would become a national rage. Then again, who would have thought that a maze game consisting of a little fellow eating dots would have a future, either.

The goal of Q-Bert is to hop down a pyramid of cubes which change color with each successful landing. When all of the cubes of the pyramid have been landed on, a new screen appears and the fun begins again.

There are a host of bad guys that pursue Q-Bert and attempt to thwart his hopping madness. The meanest is Coily, who after bouncing down the cubes as a purple ball, hatches into a snake. The only way to elude coily is for Q-Bert to hop onto a whirly disk which carries him to the top of the pyramid and lures coily into jumping off the pyramid to his destruction.

As the rounds progress, Q-Bert's speed increases, as does the speed and appearance of the nasty pursuers. At the lower levels, Q-Bert needs only to land on a cube once in order to change it to the appropriate final color. It the higher levels, up to three successive hops are needed to change the cubes color.

Q-Bert is a cartridge, can be played by one or two players, costs about $40.00 list and will run on an Atari 400/800/1200 16K machine. Q-Bert is a lot of fun to play and the graphics are very pretty.

Super Cobra by Parker Brothers

When I first booted Super Cobra I thought it was just another horizontally scrolling shoot-'em-up similar to Caverns of Mars II.

After logging over a dozen hours at the controls, I have realized that there is more to this game then the title screen.

The game scenario puts you into the pilots seat of a helicopter equipped with two weapons. You have a rapid-firing machine gun that fires directly in front of the ship. The sound of the machine gun is excellent and almost requires you to attach an amplifier and speakers to the video/audio monitor jack of the Atari 800.

Bombs are your other weapon and are released from the bottom of the 'copter. They have a good trajectory and the resulting explosion sounds very accurate. But what do you use these weapons for? That depends on the particular screen you happen to be viewing.

Throughout the game, the horizontal movement is always from left to right. Your speed is constant but you may move forward or backward within a window. At first your enemies are below. Missiles are launched vertically and tanks fire at you diagonally as you fly over the cityscape. The graphics are very colorful and add to the enjoyment of the game.

The scene changes every two screens. After the cityscape, you must fly through a cave. Sometimes the cave width is quite wide. At other times, the opening is very narrow and requires absolute concentration in order to navigate correctly. Next comes meteors in open space. The rocks appear out of nowhere and require quick maneuvering and accurate marksmenship to destroy.

This scene is followed by meteors guarding the cave openings. The meteors must be dealt with before the cave can be entered. Next comes flying saucers over a mountain range followed by flying saucers in the caves. The final screen, and the one I have yet to complete, is a maze with right angle corners. Moving forward and backward while climbing or descending is extremely challenging. This one screen alone will keep you busy for hours.

The game can be paused at any time which is quite useful given the hectic pace of the game. An especially good feature of Super Cobra is that when you have lost all of your ships and continue playing, you resume at the same level. This is a very intelligent arrangement that should be followed by more game vendors.

Parker Brothers has done a respectable job translating this arcade hit to the Atari computer. Although similar to other horizontally scrolling games, Super Cobra is much more challenging and has better graphics. It will please any shoot-'em-up fan, especially if the sound is cranked up all the way.

Super cobra comes on a cartridge and sells for about $40.00 list.

▲
▼

Report Card

any other program by typing their name. The language is not only extensible in this context but recursive as well. Simply make the last line in any program the name of the program and you have an infinite loop.

Logo has some limitations. It uses graphics mode 7 only; words are split at the end of the screen; there is no fill command, no debugging aids; pressing the SYSTEM RESET key erases all programs in memory. For all these missing features it still shines head and shoulders above its competitors.

To write justifiably about Logo would take pages. Those around me who have begun to explore it share the same enthusiasm. Atari did it right and learned from the mistakes and omissions of other Logo developers. Cost of the package varies but store-bought cost seems to be around $99.95. The school package is available in three different versions depending on whether you need to buy the three manuals. Basic school package price with only the cartridge and the reference guide is $74.95. Atari promises Logo-conversant reps on (800)538-8543. My gradebook shows Logo with an A+ and sets the standards in my classroom for others to attain. ■

Continued from 5

I think you will agree that there is much to be said for the increase in speed. One problem I mentioned is the FASTCHIP. Other differences are: You can't use SQR, COS, SIN, CLOG, EXP, and TAN with the integer run time routines. The RND() function returns integer numbers in INT. The range of INT is -32768 to +32767. Also: All DATA statements must be located at the end of the program. That and other problems are caught during compilation and displayed on the screen by the compiler, with the option to stop or continue anyway.

The end product is what really counts, and it is something to see. You can rename the file AUTORUN.SYS so that you do not have do anything. Insert the disk and boot up. DOS and the run time package will load in followed by your converted program, and then it RUNs. I have not had the greatest amount of luck trying to compile programs I already have - the compiler always seems to find something it doesn't like. But writing something after reading the chapter on what to watch out for almost always produces good results. Oh yes, the price - suggested retail is $99.95 (if you already own Visicalc, this may be your SECOND most expensive disk...). Overall, I would say that Datasoft has made a formidable effort to produce a BASIC compiler that is powerful yet also is easy enough for any level of BASIC programmer to use. ■

****
* THE GRAPHICS TABLET *
****

BY
SCOTT BRAUSE

Well here we are again, October a perfect time to work on graphics (Why? I don't know). This month we are going to take a look at ATARI BASIC'S XIO command. But before I do I want to make a reccomendation about a few books that can be very handy when working in graphics. The first one I strongly reccomend if you ever plan to do complicated graphic work in assembly language. The name of the book is THE PRINCIPALS OF INTERACTIVE COMPUTER GRAPHICS. This is an expensive book but well worth the cost. If you aren't an assembly language type than COMPUTE!'S First Book of ATARI graphics is a better book. It has a wide collection of articles from COMPUTE! Magazine.

Well, now it is time for the graphics part (remember that is what we are here for). The XIO command has many uses in ATARI BASIC. We are going to be focusing on the graphics capabilities that this command holds. Have you ever wanted to fill in that a shape box well, now you can do it. The XIO command will fill in an area specified by four points. The command format is as follows:

```
XIO 18,#6,0,0,"S:"
```

XIO, is the command name. 18, tells the computer that this will be a fill command. The #6 specifies the IOCB (Input Output Control Block) normally used for the screen. The 0,0 are used because those two commands are not needed. The "S:" is the device specifier. The S: specifier is the screen. Now in order to use the XIO command you have to set several things up. First you have to outline the fill area you do that by:

1) Enter a graphics mode

2) PLOT lower right corner

3) DRAWTO upper right corner

4) DRAWTO the upper left corner

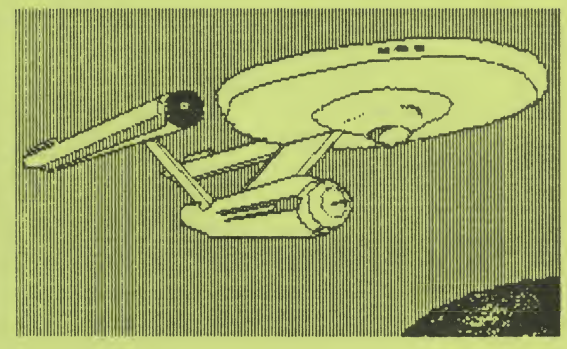
5) Use the POSITION statement to move the cursor to the lower left corner of the space to be filled.

After doing all of this you most POKE the color value you want into location 765. The color value can be any value from one to 3. After you do all this you use the XIO connand. Here is an example program that uses the above format to make a yellow shaded rectangle.

```
 10 GRAPHICS 7:REM ENTER GRAPHICS MODE

 20 COLOR 2:REM GET COLOR FROM REGISTER ONE

 30 PLOT 80,80:REM PLOT BOTTOM RIGHT

 40 DRAWTO 80,40:REM DRAWTO UPPER RIGHT

 50 DRAWTO 40,40:REM DRAWTO UPPER LEFT

 60 POSITION 40,80:REM POSITION CURSOR AT
LOWR LEFT
 70 POKE 765,3:REM LET COMPUTER KNOW WHICH
COLOR TO USE
 80 XIO 18,#6,0,0,"S:":REM XIO COMMAND
 90 END:REM END PROGRAM(THIS IS NOT NEEDED)
```

Well that's it for this month. See you next time.

●



A Dive into the Deep Blue C
By Sheldon T. Becker - J.A.C.G

While recently looking through the latest ATARI APX catalog, I noticed a program written by J. H. Palevich:The Deep Blue C Compiler. My thoughts immediately went to the possible advantage of using a C compiler on my ATARI. I could write programs on my ATARI in C, the de facto language of Bell Laboratories, and then upload these programs to the mainframe at work. This would solve the present incompatibilities I was finding with BASIC, which is not supported at work. In addition, the use of a compiler would speed up program execution. A fast phone call to the ATARI APX toll free number (800-538-1862) and the parting with $40 plus shipping via a credit card resulted two weeks later in the arrival of the program by UPS.

The Deep Blue C Compiler is an adaptation of the Small C Compiler for The 8080 microprocessor. The Small C Compiler was written by Ron Cain and published in Dr. Dobb's Journal. The Deep Blue C Compiler differs from the Small C Compiler in that a linker and an interperter were required to impliment the C language on the 6502 microprocessor. The Small C Compiler is in the public domain while the Deep Blue C Compiler is protected by copywrite. The source for the Deep Blue C Compiler is available from APX as the Deep Blue Secrets (also $40 plus shipping).

The Deep Blue C Compiler requires 48k and one disk unit. The compiler supports char, int and pointer data types, single dimension arrays, almost all of the standard C binary and unitary operators and statements like if, while, switch, do, return, else, etc. In addition, the Deep Blue C Compiler provides several BASIC like input-output, graphics , color and sound commands and also player missile commands. The Deep Blue C Compiler does not support structures, floating point, muti-dimentioned arrays and functions returning anything other than integers. The documentation for the program is brief but very good. A supplemental book such as The C Programming C by Keringhan and Retchie is necessary.

In order to generate an executable C program. the following steps are required.

First the source program must be created by using a text editor (not supplied). Next the compiler must be loaded and the source code compiled. This is followed linking the compiled code into an executable file. Loading the compilier and compiling can take several minutes even for a short program. If you write code like I do, then it will take several cycles of editing and compiling before the code is error free. These cycles become tedeous fairly quickly but I assume that programers who use micro assembles are use to it.

I programmed the Sieve of Eratosthenes prime number benchmark (BYTE-Jan. 1983) in C, BASIC and FORTH. This program generates the first 1899 prime numbers. The results were as follows for one iteration of the prime number generator: Microsoft Basic (341 seconds), C (67 seconds) and FORTH (18 seconds). All the programs were implimented using integers. The C program required about 30 seconds to compile. The results indicate that C, even with its interperter impelmentation, shows a significant processing speed improvement over BASIC.

I think that ATARI should be complimented on providing a C compiler at such a reasonable price. Other C compilers cost from $80 to $700 depending on the micoprocessor supported. The compiler has a lot of potential. An example of this is that the Deep Blue C Compilier is written in C itself. However, the lack of floating point, the fact that one can use integer numbers only up to the value of 32,767, the awkwardness of seperate editor-compiler loads and relatively long compile times limits this compiler to either the person wanting to learn the C programming language or the serious programmer. If you feel that you fall into the above two catagories, then I would recommend that you consider getting a copy of the Deep Blue C Compiler. If nothing else, this would show support of new languages for the ATARI. ▲

BLUE MAX
Reviewed by Jon Harrod-JACG

Yes folks, it's here! The game that needs no introduction, BLUE MAX! That new WW1 flying game from none other than SYNAPSE (and yes, this too has that bother some [*Obnoxious*] "New release, instant hit" sticker on it).

In this game you are flying an old WW1 bi-plane and the object is to get to the city, and while doing so sink ships, shoot down planes, shoot tanks, and blow up bridges, buildings, and other assorted targets.

The game is very well done with good 3-D graphics and fine scrolling. It features three scenarios, different skill levels, and a choice between gravity or no gravity. It also has an option where you can change between real plane controls and controls where up is up and down is down. And my favorite part is that you can strafe little cars and trucks while they drive across the bridges.

Some of the not so realistic parts of the game are the facts that before you land at an air field, you have to put your landing gear down, o.k., but on a bi-plane? And also isn't it nice of the Germans to let you have a nice, paved air strip so deep in their territory? But that doesn't affect game play much, so who's complaining?

I'm not going to say that you must have this game, because it does get boring a little fast, but I can still recomend it. □

ADVENTURES IN SOUND

By: John Rolin
From PACE Jan. 1983

Have you ever wanted to produce exact frequencies on your ATARI? As you have probably found out, the BASIC SOUND statement just won't do it: especially at the lower frequencies. The programs in this article will allow you to produce any frequency from 0 to over 64000 hertz.

I am going light on theory here it's kind of heavy; but if you are interested read Section 7 of DE RE ATARI and pages III 12, 13 & 14 of the ATARI HARDWARE MANUAL.

I will only mention here that the formula in DE RE ATARI is an approximation, which is not valid
when using the 1.79MH clock. The exact formula follows and is found in the HARDWARE MANUAL:
Fout=Fin/(2*AUDF+M) where M=7 with the 1.79MH clock. Looks heavy doesn't it?

You don't have to pay attention to this stuff because I have taken care of all of this for you. Program 1 produces exact frequencies based on coarse and fine parameters which are provided in Program 2.

I hope these programs will give you some ideas for further experimentation. How about a frequency 200 sweep to test your stereo or a touchtone frequency generator? Good luck.

```
10 REM PROG. #1 FREQ. GENERATOR
20 REM INPUT PARAMETERS CALC. BY PROG. #2
30 ?"}"
40 SOUND 0,0,0,0
50 POKE 53768,80
60 POKE 53761,160:POKE 53763,168
70 ?"ENTER FINE, COARSE PARAMETERS":INPUT
FINE,COARSE
80 POKE 53760,X:POKE 53762,Y
90 ?INT(1790000/(2*(7+256*COARSE+FINE+2)))
100 GOTO 70
```

```
10 REM PROG. #2 FREQ. PARAMETER CALCULATOR
20 REM
30 REM PROGRAM CALCULATES COARSE AND FINE
VALUES USED IN HDW MANUAL
50 ?"}"
60 PRINT "ENTER FREQUENCY DESIRED"
70 INPUT F:AUDF=(1790000/(F*2))-7
80
COARSE=INT(AUDF/256):FINE=INT(AUDF-256*COAR
SE)
44090 PRINT "COARSE= ";COARSE;:PRINT "AND
FINE= ";FINE
```

Smith-Corona TP-1 Printer

Reviewed by W. Chin - JACG

The TP-1 is the first daisy wheel printer to sell for less than a $1000. It comes in an attractive white and black case with a smoked hinged dust cover. Looking inside the case one can immediately see that this printer was designed for computer use. The moving parts look sturdy and impressive. I have continuously run the machine for several hours and it has never gotten hot. This attests to the good design of the product. The inclusion of a cooling fan located in the back of the case is just icing on the cake.

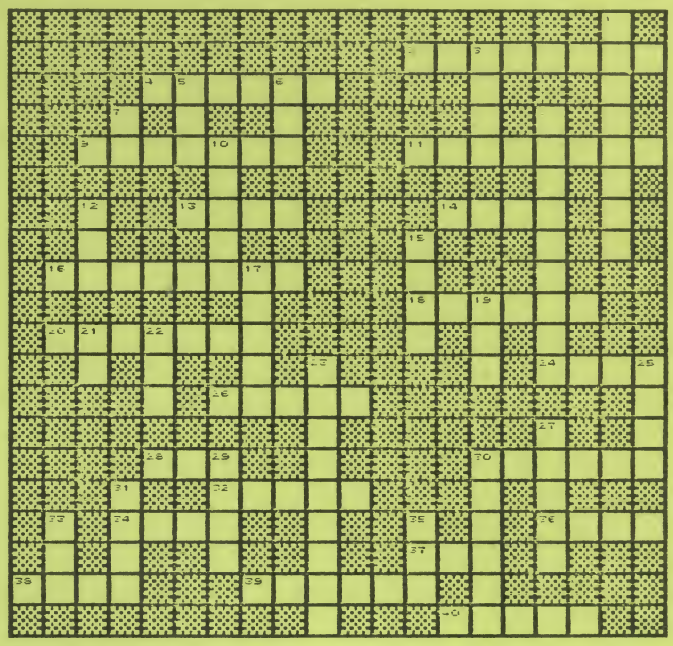
The TP-1 is available in both serial and parallel versions. There is also a choice of 10 (pica) or 12 (elite) characters per inch. It works very well with the Atari Word Processor and Text Wizard. However, with both you lose the greater and less than signs and underlining ability. As of now, I have not discovered the changes needed to solve both of these problems. With the possible availability of a driver program from Atari, this may soon be solved.

When loading paper into the TP-1, I have had to set the left edge of the paper at -3 for the Text Wizard and -4 for the Atari. By doing this I can get the one inch correspondence that each program establishes.

There are a few inconveniences with the TP-1. The first is that during a list"P" command the printer does not have an automatic return for those very long statements. The result is I lose the end of the lines on 8.5x11 paper. Another possible inconvenience is the lack of tractor feed, at least, for the moment. However, I have received questionnaires from Smith-Corona on whether I felt there would be a market for such a device. A third consideration, before buying this machine, is its inability to do graphics. Another irritation is the machine's lack of proportional spacing. I have not found it a problem. The final disappointment with the machine is its lack of being able to switch from pica to elite. You must decide when you purchase the machine which you want. At first, I thought that this would be a definite drawback. After a year of use I have not found it to be the case. All the papers that I write are required to be in pica. I find the size easier to read than elite.

The TP-1 is definitely worth purchasing if you need correspondence quality printing. The convenience of changing the cloth ribbon to the film one (for papers and to save money by doing the drafts with cloth) is a definite plus. It takes about 15 seconds for the process and your hands never touch the ribbons! (You can keep your white gloves on). It is excellent for those who write papers, and there is the added advantage of having six different types available. Unfortunately, Smith-Corona has not come out with a promised script type but the others more than make up for it. In addition, it is easy to get the TP-1 repaired since any typewriter place can repair this machine. I have had mine for over a year, and I can say, due to its simplicity, this machine should give you many years of trouble-free use.

ARI BASIC

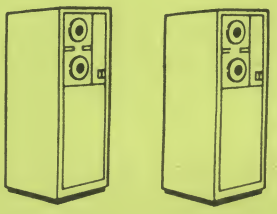


ACROSS CLUES

2. IT ADDS AND COMPARES
4. WHAT WE ALL GET TOO MANY OF AND SOMETIMES HAS A NUMBER THAT MUST BE LOOKED UP
9. AN INSTRUCTION TELLING THE COMPUTER TO DO SOMETHING
11. TO TELL THE COMPUTER WHERE TO PRINT NEXT
13. TO MAKE PERMANENT ON DISK
14. COMMAND TO VIEW PROGRAM
16. HAS SEVERAL MODES, COLORS AND SHAPES ON THE SCREEN
18. WHEN PROGRAMMING, WHAT YOU LOOK AT
20. ALSO KNOW AS A STRING
24. CATCH ERRORS, STATEMENT
26. A KET THAT STOPS THE PROGRAM DEAD IN ITS TRACKS
28. RAISES A NUMBER TO THE 0.5 POWER
30. MODE WHERE STATEMENTS ARE DONE IMMEDIATELY
32. A STATEMENT, A THE 1ST OF 3 PARTS OF ANY PROGRAM
34. LAST HALF OF TEST STATEMENT
36. AT THE END OF A CERTAIN TYPE OF LOOP
37. CLEARS THE COMPUTER MEMORY
38. MESSY WAY TO END A PROGRAM
39. THE 3RD PART OF ANY PROGRAM
40. ONE OF THE BEST FEATURES OF THE ATARI. THERE ARE 4 PARTS

DOWN CLUES

1. SPECIFY PARAMETERS OF A COLOR REGISTER
3. MY DEAR AUNT SALLY
5. LEAVE A NOTE IN PROGRAM
6. GENERATE RANDOM NUMBERS
7. MIDDLE PART OF FOR/NEXT LOOP
8. USED IN BASIC TO MAKE COMPUTER DO THINGS
10. MEANS "ON GUARD" IN GO
12. A ... NEXT LOOP
15. USUALLY FLOPPY
17. SPECIFIES WHICH REGISTER TO USE
19. TELL COMPUTER TO EXECUTE
21. A STATEMENT THAT TESTS
22. NOT NEEDED BY ATARI BASIC BUT GOOD PROGRAMMING PRACTICE TO USE
23. THE 2ND PART OF ANY PROGRAM
25. TO POSITION A DOT ON THE TV SCREEN
27. TO A TV OR HARD COPY, GIVES OUTPUT
29. WHAT A TELEPHONE DOES
30. TO MAKE A LINE IN GRAPHICS MODE
31. SPECIFIES THE INCREMENT IN A FOR/NEXT LOOP
33. ASSIGNMENT STATEMENT
35. TAKE INTERGER PART OF A NUMBER



TAPE DRIVE

```
##################################
#        J A C G                 #
#  JERSEY ATARI COMPUTER GROUP   #
#         58 DEWEY AVENUE        #
# HIGH BRIDGE, NEW JERSEY 08829  #
##################################
```


FIRST CLASS MAIL

JACG NEWSLETTER - VOLUME 3, Number 2

October 1983

J A C G OFFICERS

President: Richard Kushner
58 Dewey Ave.
High Bridge, N. J. 08829
(201) 638-8732 (home)
(201) 582-4794 (work)

Librarian: Don Ursem (Disk)
37 Clover Lane
Randolph, N.J. 07869
(201) 895-2522 (home)

Treasurer: Rick Olson
5 Starling Drive
Randolph, N.J. 07869
(201) 366-6862 (home)

Vice Pres: Dennis Kushler
45 Mountain Ave.
Summit, N. J. 07901
(201) 273-7588 (home)

Secretary: Dave Logothetis
(201) 372-5272 (home)

Programs: Richard Rospond
(201) 635-2936

Editor: Arthur Leyenberger
40 Lawrence Rd.
Parsippany, N. J. 07054
(201) 887-2861 (home)

Advertising: Herb Lehner
1375 Blair Ct.
Bridgewater, N. J. 08807
(201) 725-9394 (home)